BlockTrail: A Scalable Multichain Solution for Blockchain-based Audit Trails

Ashar Ahmad[1⋆], Muhammad Saad[2⋆], Laurent Njilla[3], Charles Kamhoua[4], Mostafa Bassiouni[5], and Aziz Mohaisen[6]
[1,2,5,6]University of Central Florida [3]Air Force Research Lab [4]Army Research Laboratory [⋆]Equal Contributors
[1,5,6]{ashar,bassi,mohaisen}@cs.ucf.edu, [2]saad.ucf@knights.ucf.edu, [3]laurent.njilla@us.af.mil, [4]kkcharlesa@yahoo.fr

***Abstract*—Blockchain-based audit trails provide a consensus-driven and tamper-proof trail of system events that are helpful in creating provenance in enterprise solutions. However, taking into account the transaction bulk generated by these applications and the throughput limitations of existing blockchains, a single ledger for record keeping can be inefficient and costly. To that end, we see an imperative need for a new blockchain design that is capable of addressing current challenges, without compromising security and provenance. Hence, we propose *BlockTrail*, a scalable and efficient blockchain solution for auditing applications. *BlockTrail* fragments the legacy blockchain systems into layers of co-dependent hierarchies, thereby reducing the time and space complexity, and increasing the throughput. *BlockTrail* is prototyped on "Practical Byzantine Fault Tolerance" (PBFT) protocol with a custom-built blockchain. Experiments with *BlockTrail* show that compared to the conventional schemes, *BlockTrail* is more efficient, and has less storage footprint.**

***Index Terms*—Audit Trails; Blockchain; eGovernment**

I. Introduction

Audit trails are important for efficient record management and provenance assurance [1], [2]. For example, government agencies are responsible for appraising properties and collecting taxes from residents [3], [4]. Starting from cities, these agencies work at various levels, including counties, states, and federation. As such, they keep track of property exchange, tax collection, permits, etc. Furthermore, these agencies have applications that generate audit trails to perform auditing and ensure system transparency. These applications continuously monitor the application's database, and generate an audit trail record upon change in the value of an object. However, due to the client-server relationship, audit trails are vulnerable to a single point-of-failure, whereby an adversary can externally and internally manipulate database and audit trails.

An intuitive solution to safeguard audit trails from single point-of-failure is to replicate them over all applications. This will raise the attack cost for the adversary since corrupting audit trails would require attacking all applications. This replication of audit trails can be achieved using blockchains, to enable secure, transparent, and immutable management of audit trails without needing a trusted intermediary [5], [6].

Current blockchain systems operate with a single-ledger shared among all system entities. The use of a single-ledger is therefore considered as a baseline model, atop which all applications abstract their services. However, the use of single ledger not only increases the storage footprint but also creates a bottleneck by preventing parallel processing. Applied to auditing, blockchains systems suffer from enormous space and time complexity, owing to the rate and size of transactions.

To address those challenges, we take a clean-slate approach towards the architecture of blockchain systems. We propose a multichain blockchain model that segregates the network into a set of layers, each capable of processing transactions independently. We leverage the hierarchical structure of eGovernment applications to facilitate parallel transaction processing and subsidized storage overhead. Moreover, the layered architecture also increases the throughput of the system by reducing congestion and transaction stall. In addition to the layered architecture, we further enhance capabilities of *BlockTrail* by using "Practical Byzantine Fault Tolerance" (PBFT) as the consensus protocol. In contrast to the existing schemes such as Proof-of-Work (PoW) and Proof-of-Stake (PoS), PBFT is energy efficient and achieves higher throughput.

A major limitation of PBFT is its lower fault tolerance compared to PoW and PoS. While PoW and PoS can withstand up to 49% malicious entities in the system, PBFT, on the other hand, can only sustain $\approx 30\%$ malicious nodes [7]. This is one of the reasons why PBFT has not been popular among applications with weaker trust models. However, specific to the requirements of our audit trail application, we take sufficient measures to equip *BlockTrail* with strong security measures in order to mitigate various attacks.

Contributions. We make the following contributions in this paper 1) We revisit the legacy designs of blockchains, and introduce a multilayer blockchain architecture that ensures higher throughput with low processing delays. 2) We present *BlockTrail*; an end-to-end blockchain solution for audit trail applications that uses the multichain blockchain model to provide secure and tamper-proof audit trails. 3) We provide the theoretical constructs of *BlockTrail* and validate its performance through experiments and simulations.

Organization. The rest of the paper is organized as follow. In §II, we present *BlockTrail*, followed by its analysis in §III. In §IV, we report experiments and evaluation, followed by related work and conclusion in §V and §VI.

II. BlockTrail Design

We begin by providing an overview of a audit log application that we use for the instrumentation of *BlockTrail*. The first step in design is the access to a large-scale audit log generation system that is currently being used by an enterprise. For this purpose, we used the services provided by ClearVillage Inc. [8], which provides software for cities and counties.

A. System Architecture

As defined previously, audit trails generated by the application are broadly associated to the exchange of property information among multiple entities at different hierarchies. This exchange of information occurs among: 1) peers (replicas)

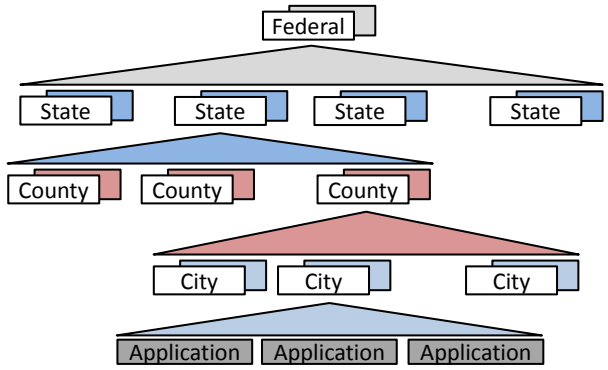


Fig. 1. Design of our multichain blockchain system that is tailored to the specifications of *BlockTrail*. Levels denote the hierarchies that keep blockchains. At the lowest level, there are applications connected to a city that emanate transactions from audit trails.

within the same city, 2) cities within counties, 3) counties within states, or 4) states within a country.

In conventional schemes of generating blockchain-based audit trails, a global blockchain is used to incorporate all the transactions. Although, this serves the purpose of secure and tamper-proof audit trails, it is not efficient and scalable. Each transaction has to traverse the entire network and get approval from all the peers. In particular, a local transaction related to a state change at the city level will require approval from all other parties in other cities that might not be relevant to that transaction. In addition to causing delays, this also limits the system throughput, since PBFT protocol serializes the transaction processing.

We argue that efficiency and throughput constraints faced by conventional systems can be resolved by partitioning the network into multiple hierarchies. As such, transactions that are specific to a group of organizations within a city must be processed locally, while the transactions related to cities within a county can be processed at the county level and stored in county's blockchain. Taking this bottom-up approach from peers within the cities to the states within a country, we obtain a hierarchical tree of blockchain system that incorporates multiple blockchains, each holding data of its corresponding set of peers. Transactions will be generated by the organizations within the cities that act as a root in the system. Each transaction will have an identifier that will determine its destination blockchain.

Using this structure, our system will be able to achieve the following features: 1) Transactions within the same branch can be processed in parallel, thereby enabling parallel processing and increasing throughput. 2) For a transaction within same branch, the approval will be required from the leaf nodes within that branch that are relevant to the transaction. This will reduce the processing overhead incurred by transactions in conventional scheme. 3) Other than transaction generation and processing, this scheme is highly efficient in blockchain queries during auditing process or for conflict resolution. In Figure 1, we show the topology of our hierarchical blockchain paradigm, and in the following, we provide the notations that capture the abstraction of our model. Let $\mathcal{L}_f = \{L_1, L_2, ..., L_s\}$ denote the country-level (federal) hierarchy that incorporates a set of all states within the country. This hierarchical blockchain paradigm can be extended from four levels to k levels, to increase the scallibility and reduce the time and space complexity.

Keeping in mind the baseline fault tolerance of PBFT, we assert that the minimum number of replicas in blockchain, at each level is $s \geq 4$. For each state in $\mathcal{L}_f$, let $L_i = \{l_1, l_2, \ldots, l_c\}$ be a set of counties in state. For each county in L_i, let $l_j = \{p_1, p_2, ..., p_d\}$ denote the number of cities that are associated with each county. Finally, for each city in l_j, let $a_q = \{n_1, n_2, \ldots, n_r\}$, be the set of peers (audit log applications), operating within the city. Given this topology, the overall size of the network S, determined by the number of audit-log applications, can be computed using.

$$S = \sum_{i=1}^{s} \sum_{j=1}^{c} \sum_{q=1}^{d} X_{qji} \tag{1}$$

Here, X_{qji} represents the position of a node within the system (identified by city, county, and state indexes). For each level, we have primary replica that executes the verification process. Specific to the design outlined in this paper, we have a primary replica for each city, county, and state in the system. Therefore, the total number of primary replicas in our blockchain system are $d + c + s$.

III. Analysis of *BlockTrail*

A. Transaction Processing and Throughput

To understand the efficiency our system with respect to the transaction processing, we use a Markovian model that broadly formulates the PBFT-based blockchain systems. To that end, we envision that the system can be viewed as a Poisson process characterized as an *M/D/1* queue at the primary replica [9]. Here, *M* denotes the arrivals determined by a Poisson process, *D* denotes the deterministic mean service time, and 1 shows that there is one server in the system. In *M/D/1* queue, as shown in Figure 2, λ denotes the mean arrival rate of the transactions at the primary replica and D denotes the mean service rate of the active replicas that collectively act as a server. From this, we can derive $\rho = \lambda/D$, which denotes the utilization of the server. If the arrival rate is less than the service rate $\lambda \leq D$, there is no queuing at the primary replica, and each transaction gets processed before the next arrival.

However, in practice, the rate of incoming transactions is usually greater than the rate of transaction confirmation [10]. Therefore, this leads to the formation of a queue at the primary replica. In PBFT, if there are a number of active replicas in the system, then the minimum number of messages exchanged to verify the transaction are $a(a-1)$. Assuming that the time taken to exchange one message in the system is t, then the total time T taken to process a transaction becomes $t(t-1)$. Therefore, as the size of network grows and the number of active replicas increase, the time taken to process the transaction decreases, and the service time of server D decreases. Some key performance indicators of *M/D/1* queue are the mean number of transactions in the system and the average wait time for each transaction. As such, the mean number of transactions L in the system can be calculated as:

$$L = \rho + \frac{1}{2}\frac{\rho^2}{1-\rho} = \lambda t^2 - \lambda t + \frac{1}{2}\frac{(\lambda t^2 - \lambda t)^2}{1 - \lambda t^2 - \lambda t} \tag{2}$$

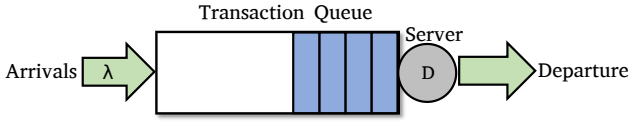


Fig. 2. *M/D/1* queue where transactions are arriving with rate λ, and there is one server that process the transactions at the average rate D

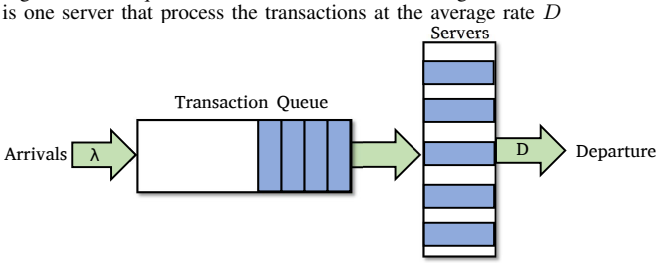


Fig. 3. *M/D/c* queue with transactions arriving at mean rate λ, and a group of servers are processing transactions with rate D.

In (2), ρ is the server's utilization, λ is the arrival rate, and t is the time taken to exchange one message. Moreover, the average wait time for a transaction in the system w is:

$$w = t^2 - t + \frac{1}{2}\frac{\lambda(t^2 - t)^2}{1 - \lambda t^2\ \lambda t} \tag{3}$$

Applied to our multichains, the total number of servers increase at each layer for parallel processing. When the number of servers increases, the number of replicas splits between the servers, thereby reducing the service time for each transaction. In such conditions, the system reflects an *M/D/c* queue. Here, c depends on the total number of replicas related to the transaction. For instance, lets assume a transaction tx_1 that is initiated between two cities C_a and C_b at time $t1$. The total replicas involved in the verification process are $a + b$. On the other hand, another transaction tx_2 is initiated at the same time t_1 among two different cities C_e and C_f, having total active replicas $e + f$. Now, these two transactions can be processed in parallel if the following condition is met:

***Condition** 1: Two transactions can be considered to be non-overlapping if their associated active replicas are unique and have no intersection.* $(C_a \cup C_b) \cap (C_e \cup C_f) = \emptyset$.

Depending on the size of replicas, each transaction will be processed accordingly. Under the assumption that at a given moment, there is a set of size c replicas that satisfy the aforementioned criteria, the system will behave as an *M/D/c* queue for transactions destined for each server. As the size of server may vary, depending on the number of verifiers involved with the transaction, the verification time and the throughput of the system may also vary accordingly.

B. Complexity Analysis

A key challenge with blockchain-based audit trails is the time and space complexity associated with the network and the blockchain size. The time complexity involves the time taken by peers to develop consensus over the blockchain state. The space complexity, involves the storage and the search overhead that compounds due to append-only blockchain design. We suggest that the multilayer architecture of *BlockTrail* can be helpful in reducing the time and space complexity to achieve faster consensus and enhance storage capability of peers.

To estimate the complexity of the system, let the total number of transactions in the system be $\widehat{t}$. Let t_d be the total number of transactions at the city level, t_c be the total number of transactions at county level, t_s be the total number of transactions at the state level, and t_f be the total number of transactions at the federal level. In (4), we show the relationship among these transactions. We assume that most transactions are exchanged at the city level, and the amount of transactions decreases as the hierarchy increases.

$$\widehat{t} = t_d + t_c + t_s + t_f \quad where\ t_d \gg t_c \gg t_s \gg t_f \tag{4}$$

$$\begin{gathered} t_d = \sum_{i=1}^{d} t_d i, \quad t_c = \sum_{i=1}^{c} t_c i = \frac{1}{2^\alpha} t_d \quad t_s = \sum_{i=1}^{s} t_s i = \frac{1}{2^\beta} t_d \\ t_f = \sum_{i=1}^{f} t_f i = \frac{1}{2^\gamma} t_d, \quad \widehat{t} = \left(1 + 2^{-\alpha} + 2^{-\beta} + 2^{-\gamma}\right) t_d \end{gathered} \tag{5}$$

1) Space Complexity: BlockTrail reduces the space complexity of system by optimizing the transaction overhead at each layer. Storage used by a conventional blockchain system is $\widehat{t} \times n$ where $\widehat{t}$ is the total number of transaction and n is the number of peers. Since $\widehat{t} >> n$ the space complexity of flat blockchain system is $O(\widehat{t})$. However, a major downside of this method is that every peer is required to maintain a log of transactions that may not be related to its application. Benefiting from the hierarchical structure of *BlockTrail* and the non-overlapping nature of transactions, we suggest that the space overhead can be considerable reduced.

In our design, a major fraction of transactions is stored at the city level. Since transactions particular within the city are only stored locally, all other cities are not required to participate or store transactions. Deriving from (5), the major fraction of city transactions can be be computed as follows:

$$t_d i + \left(2^{-\alpha} + 2^{-\beta} + 2^{-\gamma}\right) t_d \tag{6}$$

Since $t_d i$ is the dominant component, therefore, the amortized cost of storage, outside the city layer becomes negligible, and the complexity of the system approximates to the complexity at the lowest layer.

$$\begin{gathered} \left(2^{-\alpha} + 2^{-\beta} + 2^{-\gamma}\right) t_d \approx \varepsilon \\ O\left(\left(1 + 2^{-\alpha} + 2^{-\beta} + 2^{-\gamma}\right) t_d\right) \approx O\left(t_d i\right) \end{gathered} \tag{7}$$

2) Time Complexity: With respect to time, we explore two aspects of complexity namely, consensus complexity and search complexity. We show that by design, in *BlockTrail* amortized cost of search and consensus is better than the conventional blockchain.

Consensus Complexity. To achieve consensus over the state of blockchain with n replicas, $n^2 - n$ messages are exchanged. Assuming that the system receives τ number of transactions, the cost of consensus in the conventional blockchain model becomes $O(\tau^2)$. However, in *BlockTrail*, the system is partitioned into sublayers, each comprising of different number of replicas. This partitioning of the system, as shown in Figure 1, reflects a tree structure with branches depicting multiple layers. Leveraging the number of transactions at each layer and using (5), the cost of consensus in conventional(T_{conv}) blockchain and *BlockTrail* (T_{bt}) can be computed as:

$$\begin{gathered} T_{conv} = O\left(\widehat{t}^2 - \widehat{t}\right) \approx O\left(\widehat{t}^2\right) \\ T_{bt} = \left(2^{-\alpha} + 2^{-\beta} + 2^{-\gamma}\right)\left(t_d^2 - t_d\right) \approx O\left(t_d^2\right) \end{gathered} \tag{8}$$

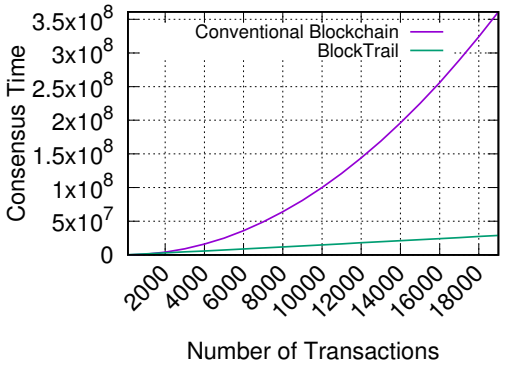


Fig. 4. Complexity Analysis of *BlockTrail*. For this analysis we used 100 peers, federal level transaction are 0.1%, state level are 0.9%, county level are 9% and city level are 90%. Consensus time is measured in microseconds.

Since $T_{bt} \ll T_{conv}$, therefore, the cost of consensus in *BlockTrail* is much less than the conventional blockchains.

Search Complexity. Similar to the space complexity, the search complexity in blockchains depends upon the number of transactions that are logged at a particular layer in the system. As such, the search cost in conventional blockchains (S_{conv}), and *BlockTrail* (S_{bt}) becomes:

$$S_{conv} = O\left(\widehat{t}\log\left(\widehat{t}\right)\right) \quad S_{bt} = O\left(t_d \log\left(t_d\right)\right) \tag{9}$$

Given $t_d << \widehat{t}$, therefore as shown in (9), the amortized search complexity in *BlockTrail* is much less than conventional blockchains. In Figure 4, we show the plots obtained by comparing complexity of conventional blockchains and *BlockTrail*.

C. Security Analysis

We perform the security analysis of *BlockTrail*. We begin by outlining our trust model and adversarial model to access the strength of *BlockTrail* against various attacks. We use our analysis to suggest possible advancements that can be made to enhance the security of PBFT-based blockchain systems.

Trust Model. In *BlockTrail*, we assume that at any level of blockchain, there are four or more replicas that process a transaction. This criteria is critical for developing consensus in PBFT blockchains, that require approval from at least $3f+1$ active replicas in the presence of f faulty replicas. Since *BlockTrail* uses a permissioned blockchain, we can assume a more trustworthy environment where peers have mutual interests and limited incentives to misbehave.

Adversarial Model. For our adversarial model, we assume a computationally-bounded adversary that controls a set of malicious replicas in the system. We assume that the adversary attains the trust of other peers and positions himself among the active replicas. If the network has n replicas and the adversary controls f replicas, then in conventional blockchain design, the value of f has to be large enough ($n - f \leq 3f + 1$) to enable the adversary to attain control over the system. If the value of f is sufficiently large, then the adversary can compromise the system by asking the faulty nodes to withhold their signatures on a given transaction in order to halt the verification. In a layered design, a major challenge for adversary is to position his replicas in a way to obtain maximum benefits with minimum effort. In the following, we discuss the possible attack options for the adversary.

D. Positioning Malicious Replicas

The attacker with f malicious replicas can either randomly position them in the network at different layers of blockchain or select a targeted layer with fewer replicas to launch a targeted attack. In this section, we will evaluate both these design choices and analyze the state of the system under attack. First, we observe the possibility when the attacker randomly allocates f malicious peers in a layered blockchain system with b number of blockchains.

The random allocation of f replicas in b blockchains can be modeled as the classical *balls-into-bins* probability problem [11]. Provided that there are b blockchains and f malicious replicas, the probability that a replica gets allocated to any random blockchain is $\frac{1}{b}$. Using this premise, we are interested in answering the following questions: 1) Probability that two malicious replicas are allocated to a specific layer of blockchain, 2) Probability that a specific blockchain has exactly p malicious replicas, where $p \leq f$, 3) Probability a specific blockchain has no malicious replica.

To answer the first question, let Alloc_i^k denote the event that i-th replica gets allocated to blockchain k, and let $\text{M}_{i,j}$ be the event that replica i and j get allocated to the same blockchain. By using Bayes' rule, we can find the probability of such an event as follows:

$$\begin{aligned}\mathbf{Pr}[\text{M}_{i,j}] &= \sum_{k=1}^{b} \mathbf{Pr}[\text{Alloc}_2^k|\text{Alloc}_1^k]\mathbf{Pr}[\text{Alloc}_1^k] \\ &= \sum_{k=1}^{b} \frac{1}{b}\mathbf{Pr}[\text{Alloc}_1^k] = \frac{1}{b}\end{aligned} \tag{10}$$

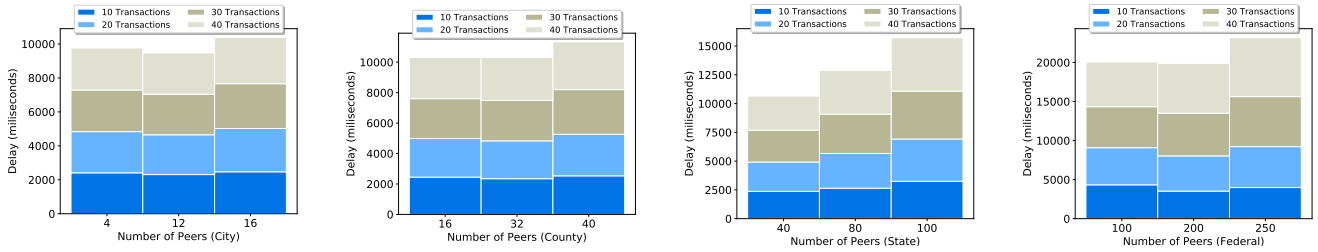
While doing the random allocation, a blockchain with more sensitive information may get exactly the number of peers that may compromise it. Let's assume that a specific blockchain b_s with p number of honest replicas cannot accommodate more than q malicious replicas. This leads to a problem raised in the second question which attempts to estimate the target blockchain gets exactly p malicious replicas, where $p \leq f$. This can be calculated by the following model:

$$\begin{aligned}\mathbf{Pr}[b_s \text{ has } p \text{ replicas}] &= \binom{b}{p}\left(\frac{1}{b}\right)^p\left(1-\frac{1}{b}\right)^{b-p} \\ &\leq \frac{b^p}{p!}\frac{1}{b^p} = \frac{1}{p!}\end{aligned} \tag{11}$$

It remains within the realm of possibilities that the attacker may not be able to position any malicious replica at any layer of the blockchain. This eventually adds to our trust assumptions of the system and may require less effort to defend against attacks. In the following, we show the probability that a specific blockchain in our system exhibits this property and contains no malicious replica belonging to the adversary. This addresses the third question above.

$$\mathbf{Pr}[\text{ blockchain gets no replicas}] = 1 - \left(\frac{1}{n}\right) \tag{12}$$

As the hierarchy of blockchain increases from city to county and eventually the federal blockchain, the security of the system also increases due to more active replicas being involved in the transaction confirmation. To enhance the security features of *BlockTrail* at lower levels of blockchain, we propose the following countermeasures.



(a) City Blockchain (b) County Blockchain (c) State Blockchain (d) Federal Blockchain

Fig. 5. Results obtained from the simulations of *BlockTrail*. Our simulation results validate the theoretical analysis. Note that as the rate of transactions increases and the number of peers grow, the consensus time increases accordingly. The consensus time is smallest in the city blockchain and it increases as we proceed towards upper levels in hierarchy. Federal blockchain experiences the same delays as a conventional blockchain.

E. Countering Targeted Attacks

In a situation where there are r number of honest replicas in a city blockchain and the attacker is able to position f faulty replicas such that $4f + 1 > f + r$, then the attacker will be able to stop transactions verification in that layer. To counter this, we propose an expected verification time window W_t which will be set by the primary replica before passing the transaction to the verifying replicas. The primary replica knows the total number of active replicas in the system and can calculate the total number of messages to be exchanged until the transaction gets verified. In this case, the total number of messages will be in the order of $(f + r)^2 - (f + r)$. If one message, exchanged among $f + r$ peers, takes t' time, then the total time taken for transaction verification will be $c \times (t'^2 - t')$, where c is an arbitrary constant set by the primary replica. Based on these values, the primary replica can set an expected time window $W_t \geq c \times (t'^2 - t')$ in which it expects all peers to validate the transaction and submit their response. Let t_{start} be the start time at which the primary replica initiates the transaction. If by W_t the primary replica does not receive the expected number of responses from the replicas, it will abort the verification process and notify the auditor.

Depending on the application's sensitivity, the primary replica can either set another optimistic value of W'_t, where $W'_t \geq W_t$, and repeat the process or it can simply abort the process and notify the auditors in application regarding the malicious activity. We leave that decision to the audit log application and its sensitivity to malicious activities. However, in our experiments, we relax the condition of sensitivity and re-submit the transaction for another round of verification. We set a new expected verification time window W'_t and wait for the response. Our choice of relaxing the condition of sensitivity is owing to the unexpected delays in the message propagation; given that our system would run over Internet. However, if the primary replica does not receive the approval of transaction the second time, it aborts the process and notifies the application.

IV. Experiments and Evaluation

We first prototype *BlockTrail* on a popular blockchain framework called Hyperledger [12], to verify its correctness and consistency with blockchain systems. However, in Hyperledger, we do not have the flexibility of applying the layered blockchain design proposed in this paper. As such, we employ the core functionality of Hyperledger including orders (primary replica), replicas, and PBFT protocol. Leveraging the design constructs of Hyperledger, we proceed with abstracting its core functionality and developing our propriety blockchain system that is tailored to the specifications of our application. In the following, we outline the steps taken to deploy our propriety blockchain system.

For experiment, we used existing logs to generate the JSON packets to generate audit trail entries. These audit trail entries are generated by the application and sent to the relevant city, county, state or federal blockchain. The primary replica notifies all concerned replicas that are associated with a blockchain and request them to validate the transaction. We generate a series of transaction for each layer of blockchain. We vary the transaction rate by increasing λ, and note the time taken by all the peers to reach consensus over it. Additionally, for each layer, we vary the number of peers and the size of transaction to see the overhead in consensus time. λ was increased from 25 to 50, and the city peers were set to 10,20,30 and 50, the county peers were set to 50, 100, 150 and 200, and the state level peers were set to 80, 160, 240 and 320. Finally, the federal level peers were set from 100, 200, 300, and 400.

We evaluate the performance of *BlockTrail* by the time taken for all the nodes to reach to a consensus over the transaction sent by the primary replica. Let t_g be the transaction generation time, and t_c be the time at which it gets approval from all active peers and gets confirmed in the blockchain. In that case, the latency l_t is calculated as the difference between t_c and t_g ($l_t = t_c - t_c$, where $t_c > t_g$). The mean values of each experiment are plotted in Figure 5. It can be observed that as the number of peers increases at each layer, the consensus time increases considerably. Also, as the rate of incoming transactions increase, naturally, the consensus time increases. As expected, the time for consensus at the city level was less compared to county and the state level.

V. Related Work

We review work on secure audit logging mechanisms and contrast them with our approach to highlight our contributions.

Audit Trails. Schneier and Kelsey [13], [14] proposed a secure audit logging scheme capable of tamper detection even after the system compromise. However, their system requires the audit log entries to be generated prior to the attack. Moreover, their system does not provide an effective way to stop the attacker from deleting or appending audit records, which, in our case is easily spotted by *BlockTrail*.

Waters*et al.* [15] proposed a searchable encrypted audit log, which uses identity-based encryption keys to encrypt audit trails, and allow search by certain keywords. Yavuz and Ning

[16] developed a forward secure and aggregate audit logging system for distributed systems, without using a trusted third party. Zawoad *et al.* presented Secure-Logging-as-a-Service (SecLaaS) for storing virtual machine audit trails in secure manner, SecLaaS ensures confidentiality of users and protects integrity of logs by preserving proofs of past logs. Ma and Tsudik [17] looked into temper-evident logs that are based on forward-secure aggregating signature schemes.

Xu *et al.* [18] proposed to use game theory and blockchain to reduce latency by moving applications to edge servers. Similarly we are using the geographical proximity to store audit logs in servers that are close to reduce latency.

Blockchain and audit trails. Sutton and Samvi [19] proposed a blockchain-based approach that stores the integrity proof digest to the Bitcoin blockchain. Castaldo *et al.* [20] proposed a logging system to facilitate the exchange of electronic health data across multiple countries in Europe. They created a centralized logging system that provides traceability through an unforgeable log management using blockchain. Cucrull *et al.* [21] proposed a system that uses blockchains to enhances the security of the immutable logs. Log integrity proofs are published in the blockchain, and provide non-repudiation security properties resilient to log truncation and log regeneration. Chi and Yai [22] proposed an ISO/IEC 15408-2 Compliant Security Auditing system using Ethereum that creates encrypted audit logs for IOT devices. Chen *et al.* [23] proposed a Blockchains based system to address shortcomings in log-based misbehavior monitoring schemes used to monitor Certificate Authorities (CA). In contrast to prior work, *BlockTrail* is implemented by extending a data access layer of the business application, which only required modification to access layer, and no other modifications.

VI. Conclusion

In this paper, we present *BlockTrail*; a multilayer blockchain system that leverages the hierarchical distribution of replicas in audit trail applications to reduce the system complexity and increase the throughput. *BlockTrail* fragments a single ledger into multiple chains that are maintained at various layers of the system. We prototype *BlockTrail* on an audit trail application and use PBFT protocol to augment consensus among replicas. We also propose new strategies to mitigate security risks associated with weak trust model of PBFT. Our experiments show that compared conventional blockchains, *BlockTrail* is more efficient with tolerable delays. In future, we aim to explore the application of *BlockTrail* beyond audit trails including IoT and health care.

Acknowledgement:. This work is supported by Air Force Material Command award FA8750-16-0301 and Global Research Lab program of the National Research Foundation NRF-2016K1A1A2912757.

References

[1] C. Wee, "Audit logs: to keep or not to keep?" in *Recent Advances in Intrusion Detection*, 1999. [Online]. Available: http://www.raid-symposium.org/raid99/PAPERS/Wee.pdf

[2] A. Ahmad, M. Saad, M. Bassiouni, and A. Mohaisen, "Towards blockchain-driven, secure and transparent audit logs," in *International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous, New York City, NY, USA*, 2018, pp. 443–448. [Online]. Available: https://doi.org/10.1145/3286978.3286985

[3] C. Ringelstein and S. Staab, "DIALOG: distributed auditing logs," in *IEEE International Conference on Web Services, ICWS, Los Angeles, USA*, July 2009, pp. 429–436. [Online]. Available: https://doi.org/10.1109/ICWS.2009.50

[4] A. Ahmad, M. Saad, M. Bassiouni, and A. Mohaisen, "Towards blockchain-driven, secure and transparent audit logs," in *International Workshop on Distributed Ledger of Things (DLoT)*, Nov 2018.

[5] M. Saad and A. Mohaisen, "Towards characterizing blockchain-based cryptocurrencies for highly-accurate predictions," in *IEEE Conference on Computer Communications Workshops, INFOCOM Workshops, Honolulu, HI, USA*, April 2018, pp. 704–709. [Online]. Available: https://doi.org/10.1109/INFCOMW.2018.8406859

[6] M. Saad, L. Njilla, C. A. Kamhoua, and A. Mohaisen, "Countering selfish mining in blockchains," *CoRR*, vol. abs/1811.09943, 2018. [Online]. Available: http://arxiv.org/abs/1811.09943

[7] G. Nguyen and K. Kim, "A survey about consensus algorithms used in blockchain," *JIPS*, vol. 14, no. 1, pp. 101–128, 2018. [Online]. Available: https://doi.org/10.3745/JIPS.01.0024

[8] ClearVillage, "Clearvillage," 2018. [Online]. Available: http://www.clearvillageinc.com/

[9] J. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, ser. Lecture Notes in Computer Science. Springer, 2001, vol. 2050. [Online]. Available: https://doi.org/10.1007/3-540-45318-0

[10] M. Saad, M. T. Thai, and A. Mohaisen, "POSTER: deterring ddos attacks on blockchain-based cryptocurrencies through mempool optimization," in *Proceedings of Asia Conference on Computer and Communications Security, ASIACCS, Incheon, Republic of Korea*, June 2018, pp. 809–811. [Online]. Available: https://goo.gl/4kgiCM

[11] P. Berenbrink, T. Friedetzky, P. Kling, F. Mallmann-Trenn, L. Nagel, and C. Wastell, "Self-stabilizing balls & bins in batches," *CoRR*, vol. abs/1603.02188, 2016. [Online]. Available: http://arxiv.org/abs/1603.02188

[12] E. Androulaki and *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *EuroSys Conference, Porto, Portugal*, April 2018, pp. 30:1–30:15. [Online]. Available: http://doi.acm.org/10.1145/3190508.3190538

[13] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 2, pp. 159–176, 1999. [Online]. Available: http://doi.acm.org/10.1145/317087.317089

[14] ——, "Cryptographic support for secure logs on untrusted machines," in *USENIX Security Symposium, San Antonio, USA*, Jan 1998. [Online]. Available: https://www.usenix.org/conference/7th-usenix-security-symposium/cryptographic-support-secure-logs-untrusted-machines

[15] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an encrypted and searchable audit log," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2004, San Diego, California, USA*, 2004. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/04/proceedings/Papers/Waters.pdf

[16] A. A. Yavuz and P. Ning, "BAF: an efficient publicly verifiable secure audit logging scheme for distributed systems," in *Annual Computer Security Applications Conference, ACSAC, Honolulu, Hawaii, USA*, Dec 2009, pp. 219–228. [Online]. Available: https://doi.org/10.1109/ACSAC.2009.28

[17] D. Ma and G. Tsudik, "A new approach to secure logging," *TOS*, vol. 5, no. 1, pp. 2:1–2:21, 2009. [Online]. Available: http://doi.acm.org/10.1145/1502777.1502779

[18] D. Xu, L. Xiao, L. Sun, and M. Lei, "Game theoretic study on blockchain based secure edge networks," in *International Conference on Communications in China, ICCC, Qingdao, China*, Oct 2017, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ICCChina.2017.8330529

[19] A. Sutton and R. Samavi, "Blockchain enabled privacy audit logs," in *International Semantic Web Conference ISWC, Vienna, Austria*, Oct 2017, pp. 645–660. [Online]. Available: https://doi.org/10.1007/978-3-319-68288-4_38

[20] L. Castaldo and V. Cinque, "Blockchain-based logging for the cross-border exchange of ehealth data in europe," in *Security in Computer and Information Sciences*, E. Gelenbe, P. Campegiani, T. Czachórski, S. K. Katsikas, I. Komnios, L. Romano, and D. Tzovaras, Eds. Cham: Springer International Publishing, 2018, pp. 46–56.

[21] J. Cucurull and J. Puiggali, "Distributed immutabilization of secure logs," in *International Workshop on Security and Trust Management STM Heraklion, Greece*, Sept 2016, pp. 122–137. [Online]. Available: https://doi.org/10.1007/978-3-319-46598-2_9

[22] S. Cha and K. Yeh, "An ISO/IEC 15408-2 compliant security auditing system with blockchain technology," in *2018 IEEE Conference on Communications and Network Security, CNS 2018, Beijing, China, May 30 - June 1, 2018*, 2018, pp. 1–2. [Online]. Available: https://doi.org/10.1109/CNS.2018.8433185

[23] J. Chen, S. Yao, Q. Yuan, K. He, S. Ji, and R. Du, "Certchain: Public and efficient certificate audit based on blockchain for TLS connections," in *IEEE Conference on Computer Communications, INFOCOM, Honolulu, HI, USA*, April 2018, pp. 2060–2068. [Online]. Available: https://doi.org/10.1109/INFOCOM.2018.8486344